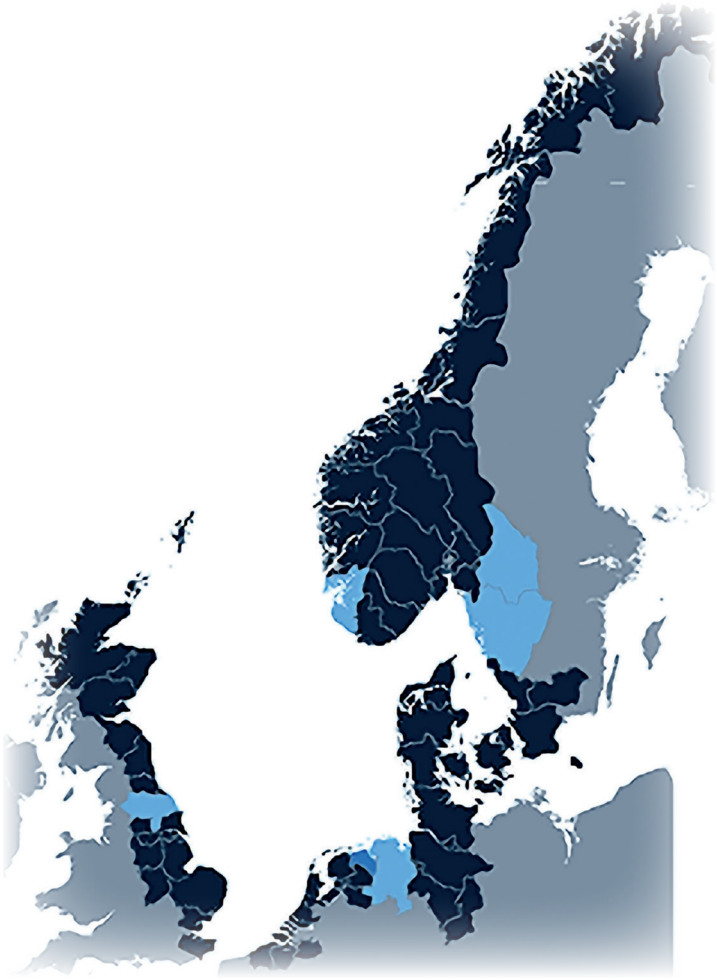
ITRACT

Improving Transport and Accessibility through new Communication Technologies

ITRACT – Best Practice Guide

Information architecture and exchange mechanisms for efficient transport concepts



Work Package 4 and 7

EUROPEAN UNION
European Regional Development Fund

Investing in the future by working together for a sustainable and competitive region

The Interreg IVB North Sea Region Programme
Investing in the future by working together for a sustainable and competitive region

Reference & Copyright

Title:	Best Practice Guide on Information architecture and exchange mechanisms for efficient transport concepts
Date:	March 2015
Commissioned by:	Hanze University of Applied Sciences
Authors:	Johan Blok and Jacob Mulder (WP 4 leader)
Address:	Hanze University of Applied Sciences PO Box 3037 9701 DA Groningen The Netherlands www.hanze.nl

Introduction

Accessibility and connectivity are essential for livability and economic growth throughout the world. Improving the accessibility of physical transport is important for achieving the social and economic inclusion of rural areas. In reality, rural areas lag behind with respect to physical accessibility and connectivity. The ambition of the ITRACT project (Improving Transport and Accessibility through new Communication Technologies) was to use ICT to create smart mobility services to improve accessibility and connectivity in rural areas.

The Digital Agenda for Europe is vital for realizing optimally accessible and connected rural regions in the North Sea Region. Moreover, collaboration between regions is essential to solve the problems of limited accessibility and connectivity in Europe. The transnational collaboration within the North Sea Region proved to be essential for realizing the mobility services within the ITRACT project, undertaken within the Interreg IVB North Sea Region Programme.

The ITRACT project started in 2012 and concluded in March 2015, developing more than 40 new ICT transport service concepts, in close interaction with users, transport organizations, transport authorities and local governments. These new smart mobility services were tested in fifteen pilots in five different rural regions in Norway, Sweden, Germany, England and the Netherlands. A novel ICT architecture was built to support the services. In a project extension awarded in 2013, new algorithms were developed to optimize the combined transportation of people and goods. These algorithms were also tested in pilots. To achieve the results, the project was divided into ten different work packages.

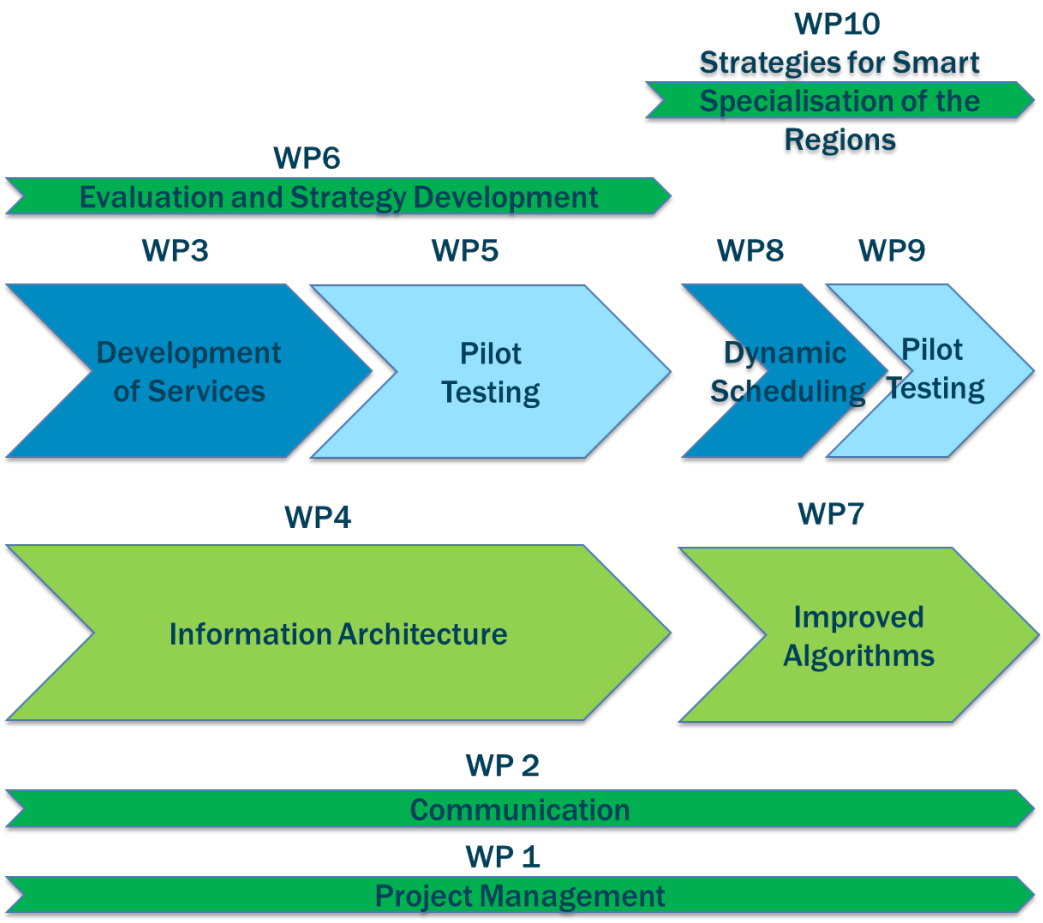
Work packages of the project

The ten different work packages were led by various project partners who collaborated in multidisciplinary and cross-border exchanges to create innovative and creative service concepts which were tested in diverse environments and regions.

General Project Activities
WP 1 Project management (Hanze University of Applied Sciences)
WP 2 Publicity and communication (University of Stavanger and Värmland County Administrative Board)

Service Development, Realization, Implementation and Testing
WP 3 Development of services and self-optimizing networks (Viktoria Swedish ICT)
WP 4 Information architecture and exchange mechanisms (Hanze University of Applied Sciences)
WP 5 Pilot testing on transport and accessibility (Jade University of Applied Sciences)
WP 7 Development and implementation of improved smart algorithms (Karlstad University)
WP 8 Dynamic scheduling and incentivizing strategies for sustainable transport (University of Groningen)
WP 9 Pilot testing on transport and accessibility (Alliance Healthcare)

Policy Recommendations
WP 6 Evaluation and strategy development (University of Groningen)
WP 10 Strategies for smart specialization of the regions (Hanze University of Applied Sciences)



This Best Practice Guide

This Best Practice Guide (BPG), '**Information architecture and exchange mechanisms for efficient transport concepts**', is the result of WP 4.

Within the ITRACT project, Work Package 4 provides an ICT platform consisting of elements that support the use of dynamic information generated by services, apps, users, sensors, etc. The platform provides apps with common functionality in the form of building blocks to solve similar problems only once. It allows apps to combine all types of transport (multimodal transport) and provides up-to-date information about the current transport situation (traffic jams, delays, congestion, etc.) potentially affecting planned trips. The platform thus provides features independent of the mode of transport and location. As such, it provides the technical means to cross the borders between modes of transport; borders that are due to the way transport is currently organized. As a result, the platform prepares the way to explore new means to organize transport.

This document is meant for ICT specialists who are building platforms for transportation support as well as for non-ICT specialists who are interested in the complex world of ICT platform realization and what difficulties can be expected. Therefore, this document provides a technical overview of the requirements, architecture and implementation of the project. Some parts, indicated with an asterisk *, are quite technical, especially those presenting the implementation and integration of the building blocks. They can be skipped by readers without technical knowledge, who are recommended to proceed with the part on management and execution of an ICT research project.

Table of contents

1 Requirements

Even today, despite technological advances, the majority of information about public and other forms of transport is still static and bound to a particular mode of transport. In other words, the information system and organization of public transport is still based on the means of communication available when the first railways appeared in the nineteenth century. The consequence is that there are huge gaps between citizens' transport needs and public and other forms of transport services offered, especially in rural areas.

Increasingly, dynamic information is becoming available – also online – for example, on delays, temporary detours, congestion and current travel needs. Dynamic information can be used to create more tightly fitting travel solutions and to update travel plans along the way when relevant changes occur. Ideally, this type of information would be used to create smarter mobility solutions that are more flexible and more fitting to current travel needs. Smarter mobility services are especially desirable in rural areas to create a better match between a relatively small number of travellers and a limited number of transport options.

The services defined in WP 3 and the planned apps of WP 5 provided input for the required functionality of the platform. These apps vary from live travel maps to stop information and ride-sharing. An ICT platform that contributes to maximum common functionality was identified by means of a systematic presentation of the planned apps and services and their needs. This resulted in the definition of nine building blocks.

The system must satisfy the following functional requirements:

- Allow for instantaneous updates to travellers about delays, etc.
- Ride-sharing: support matchmaking between drivers with plans for a trip and potential passengers
- Interactive map: display stops and current locations of vehicles on a geographical map
- Multimodal planning: generate travel plans consisting of different modes of transport such as bicycles, private cars, buses and trains
- Payment: support payments, including logging and aggregation
- User profiles: maintain user profiles, including reputations
- Dashboard: display real-time traffic information for a particular stop
- Information Storage: provide an information storage repository
- Traffic management: management of traffic
- Generic functions: small generic functionality to be used by building blocks and apps

The most important non-functional requirements are the following:

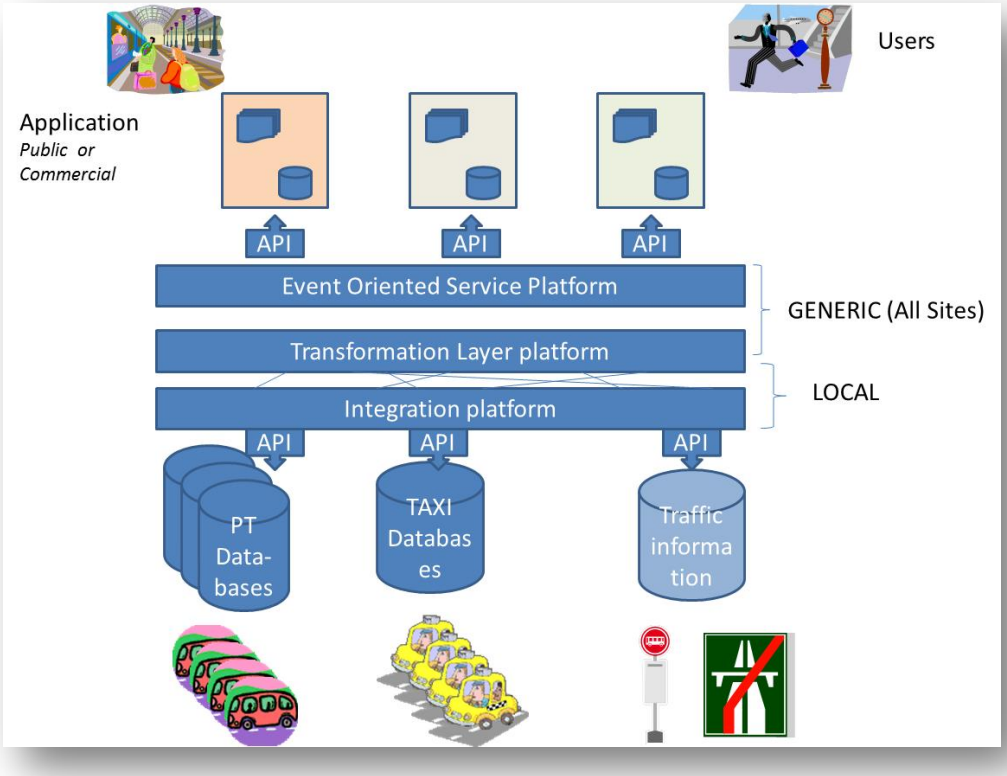
- Interface that is well documented and easy to use
- No production requirements such as fallback facilities
- Flexibility in access
- Feasible in costs (initial and recurring): preference for open source

2 Architecture

The architecture of the platform depends on the functionality that is central to all building blocks. Apart from general utility functionality, such as user profiles, the building blocks centre around transport information. Accordingly, WP 4 of the ITRACT project aimed to develop a generic distributed platform that could support a wide variety of existing and new Intelligent Transport Systems (ITS) applications. It would support a variety of scenarios in which mobile users roam and need to use different services from different transport providers in a transparent way. Since data stems from multiple transport providers, the platform must support interoperability with external systems. Thus, the aim is to design a cost-efficient flexible architecture that supports traffic data from multiple sources, and is capable of processing this information and ultimately delivering the service to consumers with fine granularity, especially when a trip includes multiple transport providers.

The ITRACT architecture assumes the availability of various open data sources and travel-planning functionalities. Typically, these include travel planners, public transport schedules, route maps, delay information, traffic information and social media platforms. Data about the transport available may stem from multiple heterogeneous data sources, whose format (for example GTFS), availability and update frequency may vary widely. An important task of the platform was to provide an interface for accessing the data in a uniform way.

A generic architecture for distributed ITS is proposed for public transport systems based on the observation that traffic data comes from different transport authorities with different interfaces and formats which creates a big challenge when a trip includes multiple transport providers. Therefore, the suggested architecture must support data from multiple sources and be capable of processing this information, delivering the service to consumers with fine granularity. This high-level architecture is presented in Picture 1.



Picture 1. High level architecture ITRACT

The ITRACT architecture is known as 'service-oriented architecture' (SOA). It enforces a separation of 'look & feel' and business logic, and consists of three separate layers:

The **integration layer** supports the collection of data from a large number of different sources such as public transport databases, taxi databases, etc. It supports both static data, such as geographical data on bus/train stops, transport schedules, etc., and dynamic data, such as changes in the schedules, weather conditions, etc. It also supports real-time data such as updates of vehicle positions.

The **transformation layer** focuses on the processing of a large volume of collected data and aligns it using a common format to allow useful operations to be performed on it.

The **event-oriented service layer** includes functionality for specifying triggers and for matching events with triggers and generating actions triggered by events. Mobile customers might, for example, receive a message when Bus No. 1 arrives at the Karlstad University bus stop.

3 Implementation*

The implementation achieved mainly consists of two elements:

- implementation of new features in the open source packages to fully implement all functionality of the building blocks
- the implementation of an environment in which the open source packages for ride-sharing and route-planning work nicely together in a scalable manner

These two elements are described in the following two sections: the latter is more technical due to the nature of integrating building blocks in a scalable manner.

3.1 Building Blocks

The building blocks were implemented as follows:

3.1.1 Ride-sharing

This building block supports matchmaking between drivers with plans for a trip and potential passengers. It was implemented by adopting an open source system called OpenRide. Proper support for multi-languages was added and several financial features were added. The GUI was replaced.

3.1.2 Interactive map

This geographical map displays stops and current locations of vehicles. It was implemented as a demo webpage for the proxy using Google Earth. The demonstrated technology was also used in apps tested in several pilot studies. It is also possible to use a geographical map without the proxy to visualize a data set. The visualization module accepts a predefined set of geographical locations with adjacent data, and displays that data on a map.

3.1.3 Multimodal planning

The multimodal planner (MMP) generates travel plans consisting of multiple modes of transport, such as bicycles, private cars, buses and trains. The implementation employs OpenTripPlanner and OpenRide.

The MMP plans the trip according to a driver's route. It aims to have a driver either at the start of a trip or the end. If the traveller can be picked up, the MMP searches for public transport stops where the user might be able to continue their trip. If pick up is not possible then available rides are searched among drivers who might be able to drop the traveller at their destination. In this case, public transport is searched in relation to any location close to the driver's route so that the passenger can be picked up at this location.

The MMP only supports trips composed of one ride-sharing part (R) and one public transport part (P). It does not allow for additional ride-share or public transport components. Therefore, *no* planned trip will be a sequence of the form R>P>R or P>R>P. The main reason for this decision is that such scenarios are unlikely to occur as people usually travel from rural to urban areas (or back). Travel between urban areas usually involves good public transport connections. The algorithm could be improved to match multiple routes if needed. However, this functionality would come with a considerable performance penalty.

3.1.4 Payment

This module provides support for payments required when ride-sharing. Due to the nature of the project – an international research project that produces pilots – the actual execution of financial transactions was left open. The module provides an interface for submitting and managing financial transactions. All registered debit transfer requests can be compiled into a batch file that can be processed by a bank. Such batch files can automatically be generated by the system.

3.1.5 User profiles

This building block maintains user profiles, including reputations to be used by ride-sharing. These are realized by extending the minimal support for user profiles offered by OpenRide. Many properties are added, such as whether the driver smokes. The functionality is used by the apps for ride-sharing.

3.1.6 Dashboard

Implemented as a webpage that periodically executes an Ajax request to the proxy server retrieving arrivals and departures for a particular location. The transit platform is responsible for an appropriate response.

3.1.7 Information storage

This building block provides storage functionality to be used by app developers to store and manage content such as images, movies, PDF files, etc. It allows the content of apps to be changed easily, functioning as an editor for the content of apps. It is also used to provide information about the system itself, such as FAQ, API-documents, etc.

Analysis of different platforms revealed WordPress to be a simple and easy to manage tool providing the required functionality. WordPress is an open source blog and content management system (CMS) based on PHP and MySQL. The plug-in architecture and template system makes WordPress easily customizable for different use cases and suitable for the ITRACT project.

The cloud-based infrastructure of ITRACT runs WordPress on three different virtual machines:

- Server running Apache, PHP and WordPress
- MySQL hosts the database
- MemCached caches the content for fast access

Such deployment on different virtual machines hosted in the OpenStack Cloud has the advantage that with increased traffic demand, more CPU and memory resources can be added according to need. This allows scalability with user and content volume.

3.1.8 Traffic management

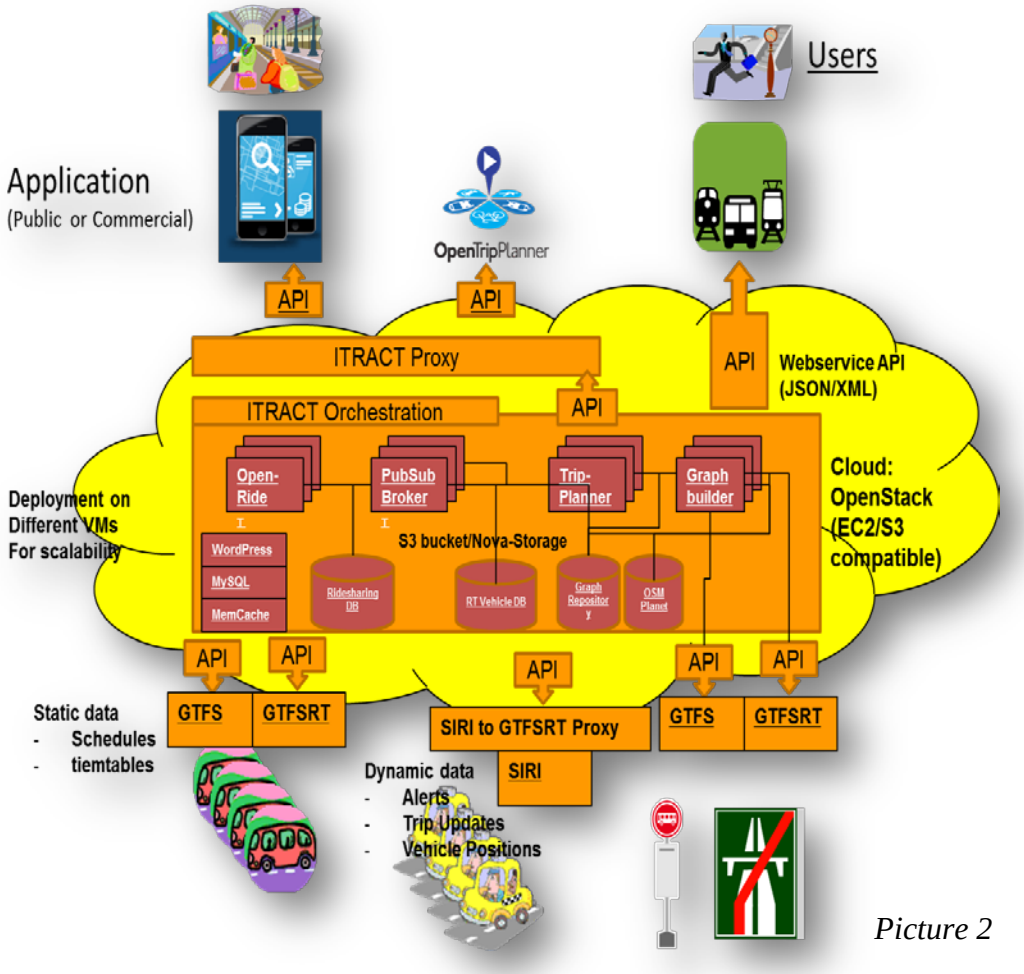
Cancelled.

3.1.9 Generic functions

- *PushMessage*: web service to push a message to a mobile app of a specific user. Enables users to subscribe to events correlated with a specific trip and receive notifications about real-time updates concerning those trips; for example, indicating to the driver that the traveller has arrived at the pickup location. As the backend has to handle potentially thousands of such updates per second, scalability is a major concern in the design.
- *Configuration*: generic storage of configuration settings as name-value pairs.
- *Security*: functions to encrypt and decrypt messages, digitally sign objects (e.g. scan of ID card), key management (connection to user database).
- *Location*: algorithms to calculate the distance between two locations and determine whether X is between two locations.

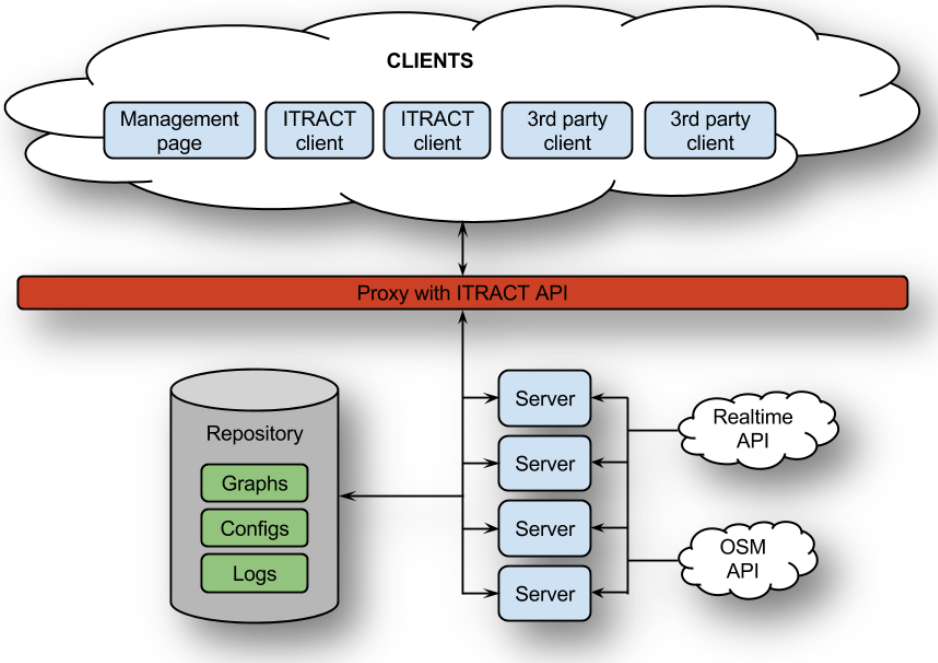
3.2 Integration

The various building blocks need to be implemented in such a way that they cooperate. The figure below illustrates how the parts of the three layers mentioned in the overall architecture are implemented in the third iteration.



Picture 2

Multiple clients with multiple purposes use the same API. A client could be an application or service for end users or used for administrative purposes (like the management page). Not all clients are developed in-house: some may be third party. The backend (in the yellow cloud) provides re-usable functionality to the clients using multiple instances of its building blocks if required. These building blocks are described in more detail below. The main part consists of route planning features for which data from public transport companies is required in the form of GTFS. Not all building blocks of the backend (yellow cloud in the architecture picture) are actually fully integrated into the API of the proxy. The following picture shows the architecture of the integrated part, which consists of a transit server combining the trip planner and graph builder. The latter is used to generate transport data (a graph) in a format that allows for an efficient planning algorithm.



Picture 3: Architecture of proxy with transit server (trip planner + graph builder)

The proxy is the master node in the cluster and contains the API implementation. The proxy receives all client requests and routes them to its underlying slave nodes. The primary function of the ITRACT backend is to supply an API that numerous types of client applications can utilize for their own purposes, each with many different users (e.g. multiple client applications, each with their own functionality and needs, each being used by multiple users and all at the same time). This kind of system needs to be both flexible and scalable and to that end the design of the backend has a computer cluster at its core.

Due to the scale at which the backend needed to operate, the distribution of the computational load is paramount for the performance and user experience. The solution to this problem involves the development of a computer cluster consisting of a centralized master node and multiple underlying computational nodes. The master node accepts all client requests and then routes these requests to its underlying computational nodes that then process the requests and compute the correct response. The cluster architecture provides the system with both scalability and redundancy.

The transit server is the slave node in the cluster. Each transit server contains transit data for a specified geographical region. API requests for that region are routed to the corresponding transit server, which calculates the correct response. The file repository contains:

- Graphs: static transit and geographical data
- Real-time configuration files: specify sources to use for real-time transit data

- Graph-builder configuration files: specifies the sources and settings on how to build new graphs
- Proxy and transit server configuration: settings regarding the deployment, such as file resource paths and update frequencies
- Logs: error logs, backup files

The implementation of the transit server is based upon the open source software OpenTripPlanner (OTP) v0.9.2: http://www.opentripplanner.org/. It provides route-planning features on the basis of GTFS data. The implementation of OTP was heavily modified for use in the ITRACT project, both in terms of new features and some significant bug fixes. The changes are now intertwined with the original OTP code and future versions of OTP cannot be used without huge amounts of work due to the high impact of the changes.

3.2.1 Cloud-based deployment

Our architecture is made out of three fundamental components: OpenStack Virtual Machines (VMs), traffic-related data sources, and consumers. The OpenStack VMs are used to deploy different ITS services related to ITRACT, such as ITRACT proxy, routing servers, WordPress, MySQL server, graph builder, etc. The fundamental objective of our system is to ensure flexible management of the available resources, such as scale down/up VMs according to load, redeployment in a new testbed with minimal effort, etc.

The deployment of an intelligent transport system delivers a number of benefits which make the system more efficient and reliable. This motivates public planners to increase the funding for transportation development. ITS is already one of the key factors stimulating economic growth in many countries. Although ITS has significant advantages, high operational costs still hinder its wide adoption. A cloud-based ITS architecture will reduce the overall system costs and hence attract more interest. Cloud-based architecture provides an efficient way of utilizing geographically distributed resources through virtualization and distributed computing techniques. The main idea behind this technology is to reduce operational costs, to increase the sharing of resources and to enable easy access to the resources through different client platforms. Some key benefits of a cloud-based architecture for ITS are mentioned below.

- Flexible Management: Cloud computing offers computing resources such as infrastructure, platform applications or business processes to consumers as a service without depending on the location. This reduces in-house operational costs by utilizing resources efficiently, and by handling dynamic resource allocation efficiently through scaling the system according to load. ITS needs to deal with a large volume of real-time information processing, especially in urban areas during peak hours. Therefore, there is a great possibility that degrading operational performance will decrease at that time. Cloud-based architecture offers a way to distribute tasks to different resources to balance the load properly. As a result, the system fully utilizes the available resources and provides the needed scalability.

- High-performance computing and storage resources: Cloud computing offers computational resources as services to utility-driven models, regardless of geographical location, in a scalable, elastic, fault tolerant and cost-effective way. An efficient ITS architecture requires the building of resources with high computational power and large storage capacity. ITS needs to collect a large volume of data through different instruments, process the data efficiently and ultimately disseminate this information on time to meet the requirements of travellers. Cloud-based architecture provides the opportunity for ITS to compute and store traffic data efficiently on clusters of distributed resources.

- Loose coupling and high interoperability among heterogeneous and distributed traffic departments: One of the important characteristics of ITS is that it includes a large number of subsystems, such as traffic control centres, different data sources and on/off-board control units. Therefore, to be efficient, ITS should have an effective mechanism that ensures that these different departments work in tandem. Cloud-based architecture has the capability to build a distributed model which is loosely coupled, standard-based and protocol independent. Moreover, this type of architecture also ensures smooth coordination among different domains of transport, and interoperability with external systems.

Although there are a large number of open source solutions for cloud platforms, OpenStack has many large, established technology and service companies who are adding wind to the sails and accelerating the legitimacy of its open source solution. OpenStack is gaining popularity because it reduces the costs of running a cloud-based architecture, specifically the licensing costs for virtualization and ongoing maintenance. Moreover, it provides a flexible way of deploying private cloud environments, and shows high-performance, scalability, elasticity and automated management of the resources. Furthermore, it uses a suite of software components which are compatible with Amazon EC2 and Amazon S3.

3.2.2 OpenStack

OpenStack currently consists of seven main projects:

1. OpenStack Compute Service Known as Nova
2. OpenStack Image Service known as Glance
3. OpenStack Identity Service known as Keystone
4. OpenStack Network Service known as Quantum
5. OpenStack Block Storage known as Cinder
6. OpenStack Object Storage known as Swift
7. OpenStack Dashboard Service known as Horizon

OpenStack Nova is the core component for building a scalable cloud platform that is mainly responsible for creating and managing clusters of virtual machines over underlying hardware. Nova consists of six different components, providing different services. Nova API provides an interface for consumers to carry out their instructions. Nova Compute manages the lifecycles of VMs. Message Queue handles the messages between different services and MySQL is used for storing all metadata. Nova Scheduler coordinates all the services and makes decisions about the resource placement based on different factors, such as processing load, memory and physical distance of the nodes. The functions of Nova Network are similar to Quantum. They are responsible for managing networks and IP addresses within the cloud platform. The Dashboard provides a browser-based graphical interface for users and administrators, allowing efficient access to and flexible management of the cloud-based resources.

3.2.3 Automated server management

The proxy has a sophisticated server management system that gathers status updates according to the following criteria:

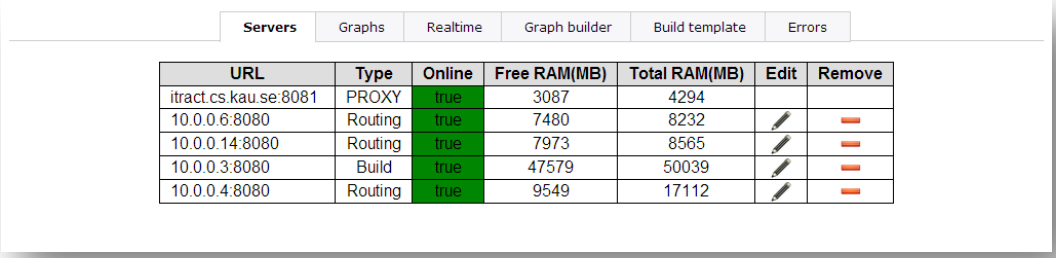
- Low manual maintenance. The management system should run by itself with low or no human input or interaction required.
- Low latency for updates. Changes in the system must be reported to the proxy instantly so that actions can be taken if needed; for example, the proxy should not send a route request to a server after it has gone offline.
- Low resource drain. The management system should not have to send or parse data if it is not necessary.

These criteria informed the design of a system in which the server helper periodically collects all the changed data from the transit platforms, aggregates this data into one single JSON string, and then sends incremental updates to the proxy.

3.2.4 Management of the cluster

Monitoring is an important aspect of a distributed system. ITRACT runs as a distributed platform on several physical servers, each one hosting multiple Virtual Machines that implement services that cooperate. It is important to gain an overview of both physical machine and virtual machine resource usage, such as CPU, memory and network, in order to troubleshoot and analyse runtime problems.

The ITRACT platform provides a web-based graphical interface for monitoring and updating the ITRACT proxy cluster (management homepage). From this page the admin can monitor and make changes to the system. These operations include, but are not exclusive: monitoring server status, monitoring graph status, removing servers, loading graphs, unloading graphs, building new graphs, creating new config files for real-time and graph-building, editing existing config files and so on.



Picture 4. Screenshot of the management homepage

Additional information can be obtained by a utility called Munin, which provides a visual way to monitor a physical or virtual host. It is based on a client server approach where a monitoring server collects statistics from munin-nodes. These munin-nodes are software agents that run on the machine which is to be monitored. Providing monitoring information at the munin-node is implemented via plug-and-play. Several plugins can be used to provide all kinds of information, such as memory consumption by the application containers in which the ITRACT components are running (Tomcat 7).

3.2.5 Routing by geographical location

Requests must be routed to a server in an efficient way. The design is based upon geo-location, where the client sends a geographical coordinate with each request. The coordinate is then matched against the geographical boundaries of each graph and routed to the server of the matching graph. If the given coordinate is inside the geographical boundaries of multiple graphs then the request is routed to the servers of multiple graphs. The response contains a list with an answer related to each graph with transit data for the given geographical coordinate. Many public transport agencies travel across borders, so if we want all the transit data from a given coordinate we can request it from both graphs of the adjacent countries.

The implementation avoids an algorithm with a linear average time, O(n), relative to the number of graphs in the proxy. In order to achieve a constant average time, the implementation uses a pre-computed geographical table representing every possible geographical coordinate on the planet, where each graph is placed in the cells that correspond to its geographical boundaries. This allows a huge number of graphs in the system without adding any delay to the routing request, since it takes the same amount of time to perform a table lookup whether the cell is empty or contains multiple objects.

3.2.6 Redundant servers for one graph

Since each graph can be deployed on multiple servers, each graph object has a list of all the servers where this graph is available. A server is chosen by a weighted randomizer function, which randomizes a value in such a way that the probabilities of the servers differ. Servers with a lower round-trip time will have a higher probability than a server with a higher round-trip time. The round-trip time is a weighted moving average that is updated for each routing request. The random nature of the function guarantees that sometimes a server with a higher round-trip time will be chosen. This prevents an overload of the server with the

lowest round-trip time, which otherwise might occur depending on the status update frequency.

3.2.7 Performance

The graphs used by the OpenTripPlanner to generate travel plans were optimized for response time (at the cost of huge memory requirements) by employing contraction hierarchies. Public transport data for densely populated countries such as the Netherlands, with many stop locations, requires hundreds of gigabytes of random access memory. Generation of the graph to update the information would take several days. To prevent time-consuming generation of a new graph, OpenTripPlanner was changed to support real-time updates to an existing graph.

New pilots in Work Package 7 of the ITRACT project involve the evaluation of logistics models, deciding where to locate facilities given the location of customers, the availability of transport and the preferences of customers. This requires the capacity to process large batches of travel requests. Support was added to OpenTripPlanner.

3.2.8 Suggested improvements

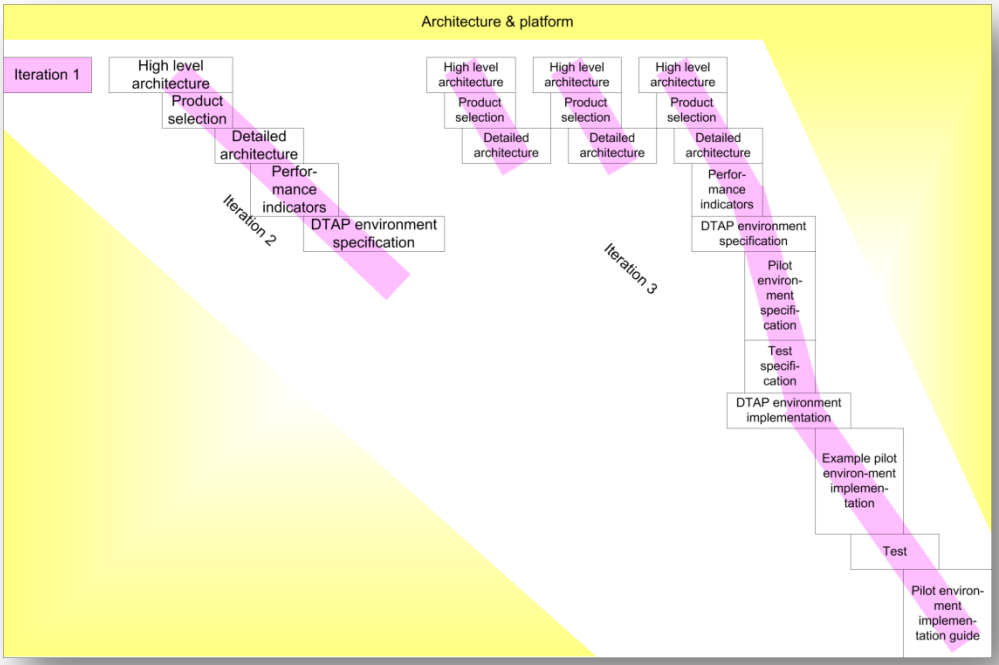
The proxy and transit server in particular provide an excellent case to pursue further research on route planning algorithms, data processing and distributed event architecture. It also allows students to gain experience in contributing to a larger system. The most important fields for improvements are:

1. Minimize memory requirements for graphs. The memory requirements for loaded graphs are very high in OpenTripPlanner. This part of OTP has not yet been optimized.
2. Optimize graph building. The OTP graph-builder is very slow, the main reason being that it is single threaded. It downloads OSM data and then processes it, then downloads more OSM data and processes that, all in one sequential thread. Making this process multithreaded would speed it up significantly.
3. Calculate memory requirement from graph file size. When a graph is loaded the Proxy chooses the server with the most free memory, but never checks to determine whether there is enough memory. It is possible to load a graph that requires more memory than the machine actually has. The memory requirement needs to be calculated.
4. Automatically build a new graph when new GTFS data is available rather than waiting for the graphs to become outdated. Check the GTFS sources to determine whether there is new data. If so, rebuild the graph and then redeploy it on all machines, one at the time.
5. Link GTFS-RT vehicle position data with static data. When a vehicle reports which route or trip it is taking, link that data with the static GTFS data and provide a next stop for a vehicle, next vehicle due at a stop, destination display on the vehicle and more.
6. Implement graph pool. Rather than specifying which server should have which graph, just ensure that this graph is loaded on X number of servers. When a server goes down, the graph is automatically loaded onto another one.

4 Process & Iterations*

As a research project, the platform demonstrates techniques and innovation rather than establishing a stable production-ready infrastructure. Three iterations were planned to allow exploration of a diverse range of approaches.

The third iteration implemented the platform by adjusting and combining the open source projects discovered, consisting of a trip planner, a project providing real-time information and a project for ride-sharing (OpenTripPlanner, OneBusAway, OpenRide). The implementation of the final and third iteration was presented in the previous section. This section describes the previous iterations, providing an overview of the options considered during the development of the platform.



Picture 5

4.1 First Iteration

The first iteration was planned as a short exploration. From the very beginning it was clear that the huge amount of transport data would require distributed computing. Accordingly, two common elements in distributed computing, namely the enterprise service bus (ESB) and multi-agent technology, were explored. Additionally, the first iteration consisted of an evaluation of the Dutch data sources for public transport. The research was executed by groups of students during a study unit of ten weeks.

The need to allow a quite diverse range of services to cooperate requires a middleware solution. The most promising candidates, namely an enterprise service bus (ESB) and multi-agent systems, were explored. Apart from research into candidates for the architecture of the platform, the first iteration contained an investigation into a data provider for public transport in the Netherlands.

4.1.1 ESB

The WSO2 product suite was the most promising candidate for an ESB because of its extensive feature set and availability as open source. WSO2 was set up and attempts were made to create a web frontend which could access a number of existing services. This resulted in the following conclusions about WSO2:

1. Difficult to set up due to the business model: the software is free, but support and hosting are not. The activity of the online community is quite small and there is virtually no technical documentation, so support is definitely required.
2. Hosting the WSO2 suite is an option, but too expensive for ITRACT.
3. WSO2 turned out to be a rather heavy user of memory and processing power.

After thorough research, the group decided not to use WSO2, relying on the Microsoft suite instead. The team used Atlassian and AppHarbor as the development environment to create a demo application consisting of a web frontend and a mobile app using Microsoft software (.NET/WindowsPhone). This demo plans a trip, shows all stops and displays this information on a map. The demo also shows real-time disruptions and the location of buses.

4.1.2 Multi-agent technology

With the requirements of scalability and a heterogenic landscape for service in mind, an agent-based solution seemed a qualified candidate for the platform. The agent implementation JADE was selected as a candidate for the ICT platform for ITRACT.

The group of students who undertook the research focused on public transport on the Dutch island of Texel. They created a demo with both a web frontend and a mobile app. It shows the public transport stops in the neighbourhood of a GPS location and showed the real-time location of buses, trains and ferries. The research group partially faked the actual public transport data in order to focus on the agent technology.

They concluded that agent technology is a suitable candidate for ITRACT. The initial setup is very important and requires sufficient knowledge of agent technology, which was not available in the group of students. As a result, establishing a proper initial setup of the agents turned out to be quite difficult and time consuming. However, the extendibility and scalability are quite good.

4.1.3 GOVI

GOVI is a system that provides real-time information about Dutch public transport. The team created a demo with a web frontend and a mobile app to display bus stops in the vicinity and the real-time location of buses on a map. The team discovered that the actual delivery of data by GOVI did not fulfil expectations based on the documentation available on various sites. The data lacked both completeness and quality. Changes in stop locations, in particular, resulted in confused data.

4.2 Second Iteration

The aim of the second iteration was to fully implement the platform employing ESB technology. Difficulties with implementing the chosen ESB suite and the discovery of several open source projects led to an early end to the second iteration. The planned deliverables of the second iteration, such as performance indicators, test specifications, etc., were cancelled.

Multi-agent technology was considered but rejected because it would add several additional risks to the project:

- multi-agent technology is still seldom used in practice (relatively difficult to control a running environment)
- multi-agent technology requires a lot of knowledge and experience, which the participants did not possess.

The first iteration showed the WSO2 to be a fascinating product. Unfortunately, it also proved very difficult to run and operate without quite expensive support by the commercial WSO2 corporation (http://www.wso2.com). Although this constituted a rather serious impediment to the project, it was still felt that the software suite should not be overlooked, primarily because of its unique features, such as support for event-driven architecture and open source advantages. R&D, including different hosting, that is, not purchased from WSO2, was thus to be continued.

4.2.1 Process

At this stage of the project, apps and pilots of other work packages were already being tested. Transnational meetings brought a series of observations and findings to light.

An open source project called ‘One bus away’ was discovered. This bus-tracking system, based on GPS location, is already up and running. It offers some functionality similar to ITRACT. The deployment of WSO2 was subsequently interrupted. It is now on hold while the usability and applicability of ‘One bus away’ is investigated. In Germany, a university is using ‘Ride-sharing’. In addition, Mitfahrzentrale is already an existing phenomenon in Germany (www.Mitfahrzentrale.de).

With these and other new suites in mind, it is recognized that all of the necessary building blocks can now be fully designed and specified. The single most important remaining step to accomplish is the integration of all parts, without the further use of WSO2. Consequently, all the other steps in the fixed developmental sequence of Iteration 2 have become obsolete – in other words, all those aspects following the detailed architecture of Iteration 2.

5 Lessons Learned

Apart from the technical knowledge concerning distributed transport systems described above, important lessons were learned concerning the execution and management of the process of developing such a system in the context of a research project.

- **Summary conclusion for non-ICT stakeholders**

ICT is a complex part of any project. Communication between ICT and non-ICT partners is key. Interaction on functionality and usability should be regular and iterative. Non-ICT stakeholders should therefore stay abreast of the ICT developments and keep the communication going so everyone has a common understanding of what is needed and is able contribute to the overall goal. Expectations of every stakeholder must be clear and managed. Defining the necessary functionality and usability and monitoring the development of this functionality and usability is not only an ICT responsibility, but also that of the clients and users.

There is a lot of open source software in the public domain that can be reused to create smart mobility applications. Using open source as building blocks for new applications is attractive because it minimizes development costs and time. However, make sure the open source software is compatible and up-to-date with the functionality needed, because adjustments are complex and not all open source software is actively maintained and kept up-to-date. The ICT partners should explain the implications of reusing open source.

The use of open source software rather than vendor-owned software may make the ICT landscape difficult to maintain. Make sure that the project has the right ICT people on board to handle open source software and to maintain the overall software architecture when it is delivered.

- **Maintenance and security**

Maintenance and support activities comprise both a helpdesk offering support to current users and can offer an assessment and improvement service for developing new functionality. The helpdesk consists of both the implementation of supporting ICT tools and the development of the organization surrounding it, including the design and implementation of the necessary procedures specified.

Operation of a full development, test, acceptance and production (DTAP) environment costs time and money, which comes on top of the time and money involved in developing the necessary features (e.g. extensive detailed test specifications are required). Within the ITRACT project, there was not sufficient time or budget to allow for a full implementation of the DTAP process in all regions. A limited version was set up to support the pilots. However, when planning and budgeting for the live implementation of new applications, one

should look for an organization that is able to maintain the applications and the underlying architecture and budget for the costs involved.
This organization should also be tasked with managing the security of the data involved. While the use of personal data increases the usefulness and added value of the applications, it also introduces vulnerability with respect to security.

- **Regulations for transport data**

The countries involved in ITRACT all have their own national laws and regulations. For this reason alone, it is inevitable that all public transport information from one country needs to be translated, as it were, from its own national format and/or standard to that of the other country.

For example, the United Kingdom has defined and developed a completely separate standard from the rest of the participating countries. This standard, called the TransXChange, has been set out in an Act. The Act is binding for all public transport companies. As public transport is fully privatized in the UK, and all companies adhere to the TransXChange standard, this is an achievement of the highest level.

The Netherlands uses GTFS (General Transit Feed Specification), which defines a common format for public transportation schedules and associated geographical information. GTFS allows public transit agencies to publish their transit data, and developers to write applications that consume the data in an interoperable way. GTFS is a widely used standard. ITRACT has adopted the GTFS standard but has not made it publically available.

Another example is Germany's Deutsche Bahn. The organization is the only official body in Germany that is legally permitted to decide on itineraries and timetables. This prevents others, such as Google, from using it to their own advantage.

In all three iterations, intellectual property law proved to be very important when it came to licensing. In Iteration 1 the use of certain purchased products, such as Microsoft, IBM and Oracle, became unfeasible, as the licensing became excessively expensive and any further use became legally impermissible. Having embarked on WSO2 during Iteration 2, it became apparent that although the product was freely available – including download – it proved to be so laborious and so difficult to install that it would have been necessary to commission the WSO2 company to solve the problems.

As the research project only delivered pilots, laws prohibiting the provision of services similar to taxis did not damage the project, as recently experienced by the new taxi service Uber.

Several projects similar to ITRACT undertaken elsewhere in the world were also discovered. The products of each of these projects appeared to use free intellectual property and as such they could be used by ITRACT and hence constituted Iteration 3.

- **Research versus production**

As a research project, ITRACT merely realized pilot systems. As a result, there remains no uptime guarantee, nor any backup system. Moreover, after running the pilots, there is no assurance that continued use of the central system will occur. However, third parties, especially those neither acquainted with the software projects nor with the research project, tended to have much higher expectations, especially when the pilot proved successful.

All parties involved extensively discussed the various ideas and concepts developed for the building blocks, the apps, the pilots and tests. This ultimately resulted in a change of plan, as a number of regions chose not to wait for ITRACT to deliver the Central Platform and its Building Blocks. Instead, they opted to involve external parties (local ICT companies) to build the apps and immediately go live.

The innovative ideas that were devised during the transnational cooperative process of the project were considered so promising that the cascade-like structure of Iteration 2 was cut short. Subsequently, the regions started to implement real, live local solutions based on these ITRACT ideas and concepts, using the resulting apps and systems as the basis for their pilots. Paradoxically, the fact that the majority of the regions will not use the central system can be considered as one of the great successes of ITRACT, insofar as the success of any research is mainly determined by the adoption of the ideas it develops.

- **GTFS data**

Inherent to the use of the open source suite ‘Open Trip Planner’ is the use of GTFS. The public transport data were uploaded into the system by GTFS format. This decision, however, has had a rather unfortunate consequence due to the fact that GTFS is almost too precise. When travelling from A to B, there are two bus locations, that is, a point of departure and a point of arrival. At either one of those points, there may be other stops for different buses within a few metres. With regard to the Dutch data for the OV-bureau this caused a problem: the unnecessary calculation of routes between all of these stops.

In England, where TransXChange is used, the centre point of such multiple bus stops will be taken when calculating a route. The Dutch system, with its precision, causes problems in Trip Planner by including every bus stop nearby and calculating as many routes. This approach results in data contamination, also labelled ‘garbage in garbage out’.

With Open Trip Planner failing in this respect, the open source suite was subsequently adjusted to properly cope with the data, without losing efficiency. It now excludes all other bus stops within a radius of 50 metres of the requested points of arrival and departure, and all necessary stops in between, in order to produce a correct calculation from A to B.

- **Dependency management**

WP 4 employed a number of off-the-shelf software suites which were built externally. Some are now outdated and no longer maintained. For technical reasons, one of the software suites, Ridesharing (Open Ride), had to be reinstalled on a server. Problems arose because there had been multiple updates since the original version had been built: not only in the operating system, but also in the database, the corresponding JavaScript and other technical environments. A student group investigated and studied the problem and eventually came to the conclusion that additional help was required. An external specialist was hired to tackle the issue.

When a project depends on external software packages, they must be maintained. In the event of neglect, much more time and many additional resources will be required to keep everything functioning properly. Therefore, the maintenance costs might outweigh the costs of developing a new implementation. In the context of a research project and an educational institute, it could also be argued that development activities rather than the maintenance of a project result in more worthwhile knowledge, since the development of functionality usually leads to increasing abilities and more profound knowledge of technical topics, while maintenance merely increases knowledge of particular versions of frameworks.

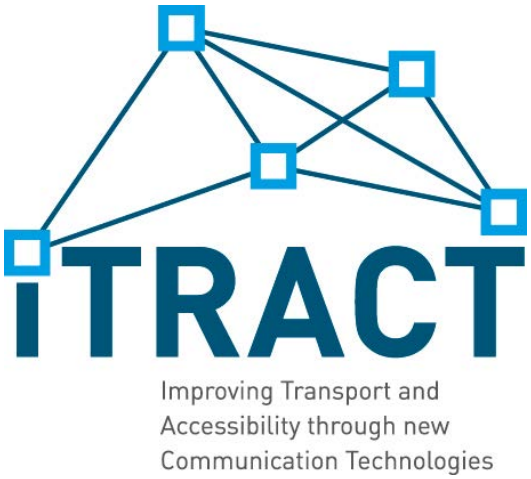
When considering whether to use an open source software package, the following must be taken into account:

- the current activity on the project
- the (in)compatibility of its dependencies and deployment requirements with other parts of the project
- source codes and the deployment of projects should be kept separate to guarantee a loose coupling of components.

- **ICT versus the rest of the world**

During the ITRACT project a gap between the ICT and the non-technical work packages was felt by many. People working on both sides had difficulties understanding each other and finding common purpose. An important lesson learned is that a mediator who is familiar with both worlds can ensure better communication between and the alignment of the goals of future users of ICT and the ICT developers. The Agile software development methodology is designed to offer a means to bridge the gap between technical and non-technical project members, which became apparent in other concurrent projects undertaken by ITRACT partner Hanze University of Applied Sciences.

Every Best Practice and Lesson Learned, with 20/20 foresight and 20/20 hindsight respectively, will offer one specific reference with regard to its content: ITRACT website, www.ITRACT-project.eu



www.itract-project.eu

Contact

Hanze University of Applied Sciences
Jacob Mulder
Phone: +31 641 424 634
E-mail: j.mulder@pl.hanze.nl
www.hanze.nl

Lead Beneficiary
Hanze University of Applied Sciences
www.hanze.nl

Drs. Theo Miljoen
Phone: +31 683 099 791
E-mail: t.a.miljoen@pl.hanze.nl

EUROPEAN UNION
European Regional Development Fund

The Interreg IVB North Sea Region Programme